# FACE RECOGNITION SYSTEM USING LOCAL BINARY PATTERN BASED ON ONE SAMPLE PER PERSON PROBLEM

**THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF**

**MASTER OF SCIENCE**
**IN**
**COMPUTER SCIENCE**

**By**
**Bipul Kalita**
**Roll No : 137/16**

**UNDER THE GUIDANCE**
**OF**
**Dr. Pranamika Kakati**
**Assistant Professor**
**(contractual)**
**Gauhati University**

**DEPARTMENT OF COMPUTER SCIENCE**
**GAUHATI UNIVERSITY**
**GUWAHATI, INDIA**
**JUNE –2018**

GAUHATI UNIVERSITY

**DEPARTMENT OF COMPUTER SCIENCE**

**Gopinath Bordoloi Nagar, Jalukbari Guwahati-781014**

# DECLARATION

I ***Bipul Kalita***, Roll No ***137/16***, a M.Sc. student of the department of Computer Science, Gauhati University hereby declares that I have compiled this thesis reflecting all my works during the semester long full time project as part of my M.Sc curriculum.

I declare that I have included the descriptions etc. of my project work, and nothing has been copied/replicated from other’s work. The facts, figures, analysis, results, claims etc. depicted in my thesis are all related to my full time project work.

I also declare that the same thesis or any substantial portion of this thesis has not been submitted anywhere else as part of any requirements for any degree/diploma etc.

Bipul kalita

Date: 26/06/2018

GAUHATI UNIVERSITY

**DEPARTMENT OF COMPUTER SCIENCE**

**Gopinath Bordoloi Nagar, Jalukbari Guwahati-781014**

Date:

## CERTIFICATE

This is to certify that **Bipul Kalita** bearing Roll No: ***137/16*** has carried out the project work ***face recognition system using local binary pattern based on one sample per person problem*** under my supervision and has compiled this thesis reflecting the candidate's work in the semester long project. The candidate did this project full time during the whole semester under my supervision, and the analysis, results, claims etc. are all related to his/her studies and works during the semester.

I recommend submission of this thesis as the project report as a part for partial fulfillment of the requirements for the degree of Master of Science in Computer Science of Gauhati University.

Dr. Pranamika Kakati
Assistant Professor
(contractual)
Gauhati University

GAUHATI UNIVERSITY

**DEPARTMENT OF COMPUTER SCIENCE**

**Gopinath Bordoloi Nagar, Jalukbari Guwahati-781014**

# External Examiners Certificate

This is to certify that ***Bipul Kalita***, bearing Roll No ***137/16***has delivered his project presentation on ***29/06/2018*** and I examined his thesis entitled ***face recognition system using local binary pattern based on one sample per person problem*** and recommend this thesis as the project report as a part for partial fulfillment of the requirements for the degree of Master of Technology in Computer Science of Gauhati University.

(External Examiner)

# ACKNOWLEDGEMENT

I feel immense pleasure in expressing my sincere thanks and gratitude to my project guide Dr. Pranamika Kakati, Assistant Professor, Department of Computer Science, Gauhati University for her valuable guidance and constant encouragement throughout the entire project work entitled "**face recognition system using local binary pattern based on one sample per person problem**". Without which this study would have never be possible. I express my honest salutation and gratitude to all the teachers of the department, for their guidance.

Finally, I really thank my friends and family for their continual support and encouragement.

Bipul Kalita

M.Sc. 4th semester
Roll No: 137/16
Reg No: 034386 of 2013-14
Department of Computer Science
Gauhati University.

## **ABSTRACT**

In a face recognition system the system has to identify an unknown face based on it’s stored database. There are several techniques used in face recognition system. However most of them needs large databases of faces per one person to identify a face. In this project using Local Binary Pattern as features and Euclidean distance as classifier are used. A classifier is a system which classifies object. It can classify a face, this classifier is build from existing data so it’s called supervised learning. In this project the aim is to identify an unknown person by its face using less number of existing images. For the experiment different datasets have been used and obtained the accuracy of the  recognition system, Also later possible matching faces are computed.

## PROJECT DESCRIPTION

| | | |
|---|---|---|
| Title | : | Face recognition system using local binary pattern based on one sample per person problem |
| Category | : | Image Processing |
| Project Guide | : | Dr. Pranamika Kakati |
| Aim | : | To recognize an unknown face of a person |
| Language | : | C++ |
| Library | : | OpenCV 3.4 |
| Platform | : | Linux |
| Duration | : | 1 Semester |

<u>CONTENTS</u>

# CHAPTER 1. INTRODUCTION AND OVERVIEW

## 1.1 Introduction:

The face recognition system has two parts : face verification and face identification (simply recognition)
but at the beginning stage the faces are extracted from images and preprocessing them. Then comes recognition part, it includes feature extraction and the matching. The face recognition system can be classified in to following parts mainly -

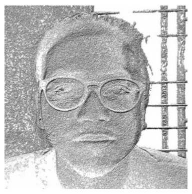
- **Holistic Methods**: face image is used as the raw input to the system. Example : Local Binary Pattern(LBP), PCA-based technique etc.
- **Local Feature-based Methods**: features such as eyes, nose and mouth are extracted and then their locations and appearances are the input to system. Example : Elastic Bunch Graph Matching (EBGM).

In this project Local Binary Pattern (LBP) is used for feature extraction.

## 1.2 Local Binary Pattern(LBP)

[1]It was first described in 1994. It is a simple and very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number. LBP operator has become a popular approach in various applications. The most important property of the LBP operator in real-world applications is its robustness to monotonic gray-scale changes caused, for example, by illumination variations, example :

Image 1 (dark)


LBP image of 1

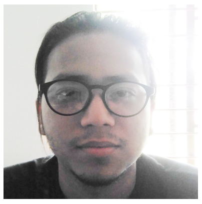
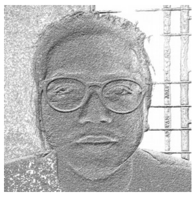


Image 2 (bright)


LBP image of 2

Also LBP computational cost is low which makes it possible to analyze images in challenging real-time.

# CHAPTER 2. LITERATURE REVIEW

Face recognition based on one sample per person problem:

In a survey[2], several feature extraction methods are used to result which method is best for one sample per person face recognition problem, and they have found LBP as it gave more accuracy.

In the research paper [3], they have used a tree based data structure which divides the face image to some sub-grids and then extracted LBP features. For different types of grid different accuracies are obtained.

[4] contains works using LBP where image is partitioned into 7x7 grid and then calculated histogram and concatenate them, then the recognition result was obtained by using chi-square distance from histograms of test images and training images. The results was surprising, using only one training image per person the system gave 85.81% accuracy.

# CHAPTER 3. THE PHASES OF THE PROJECT

## 3.1. Preparing face images:

For training datasets i have taken equal number of images for each person. Then, I extracted faces and resizes them to 100x100 pixels. Images of each person have been stored in separate folder.

Similarly for 1 face image to be tested, the same face extraction and resizing is done.

Accessing images in the program:

Two text files are created, one contains locations(file path) of all training images and other one contains path to test images.

**3.2. Finding LBP(local binary pattern ) from image:**

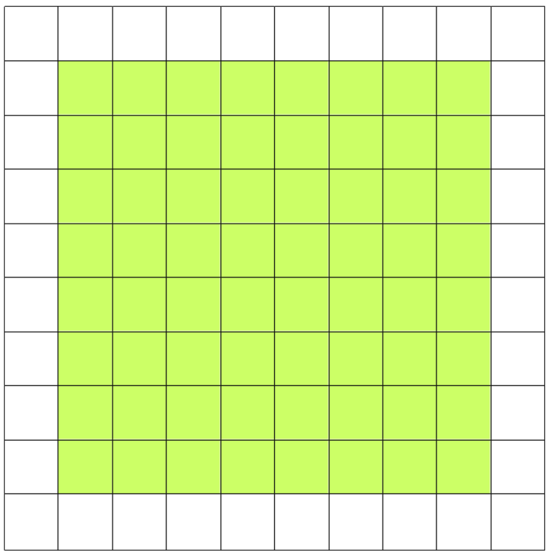
The size of the image is 100x100 = 10,000 pixels.

Then next, the image is converted to gray scale image. Where each pixel has a value between 0-255, 0 for pure black and 255 for pure white.

Next the image is partitioned into 10x10 blocks,
where each block contains 10x10 = 100 pixels,
then from each block LBP is calculated and stored in an array with associated block position as index.

Calculating LBP for a block:

To calculate LBP for a block we have to select centers where each center has 8 neighboring pixel and then calculate LBP for each center.



10x10
A block

🟩 center

□ Discarded to be a center

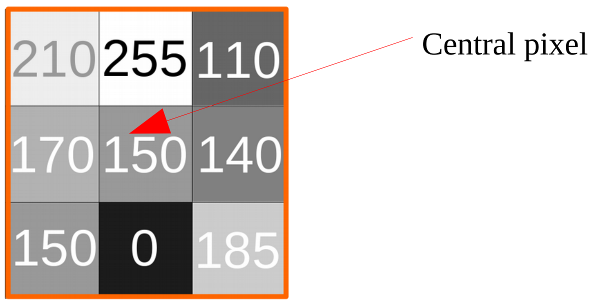
Total centers for one block = 8x8
= 64

Total centers for 1 image = 10x10x64
= 6400

i.e. 6,400 LBP features for one image

Calculating LBP for a center:

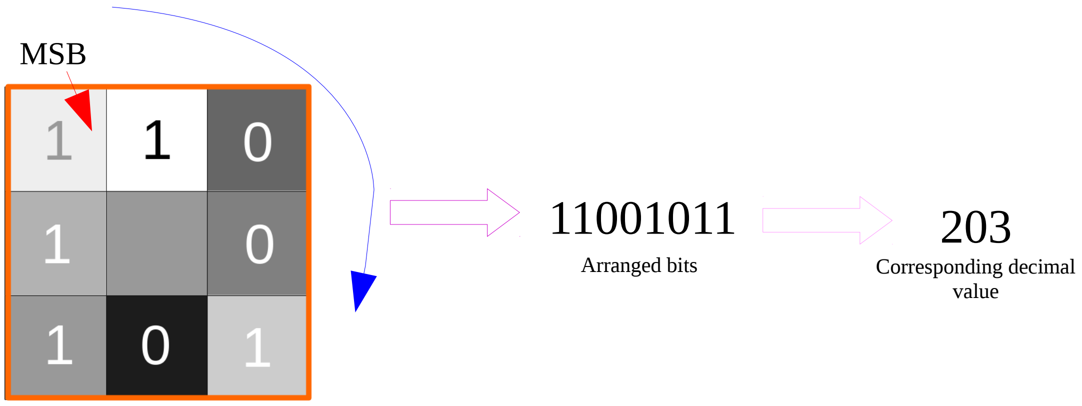
To illustrate how to find the LBP, the following picture with pixel intensities is considered.



Selected pixel as center to calculate LBP with 8 neighboring pixels

Now, compare each neighbor pixel with center, if the neighbor pixel has higher or equal intensity than central pixel then consider 1 for that position else consider 0.

Then, we will have 8 numbers for 8 neighbors pixel, each one is either 0 or 1, so arrange those 8 bits by selecting any one bit as most significant bit(MSB) and then select other bits by moving either clockwise or anticlockwise. It is noted that, selecting bits can be in any order but the same order must be maintain for all centers in every block in every image. Then calculate the corresponding Decimal value of that 8 bit binary number and then store it in the array in corresponding center location as index.



After comparing pixel intensities

The algorithm to find LBP of a central pixel C as follows:
(consider that the pixel C has 8 neighbors as we selected)

```
begin
step 1: initialize i=7,result=0
step 2: for each neighbor pixel p of C do:
step 3:         if intensity(p)>=intensity(C) then
step 4:                 result = result + $2^i$
step 5:                 i--
step 6:         end if
step 7: end for
end
```

### 3.3. Calculating Histogram Data from LBP features:

After calculating LBP features of an image, we have now 6,400 LBP features from 10x10 = 100 blocks, now each block contains 64 LBP features which is 8x8 features in a block.

Now from that 8x8 features from each block we have to calculate histogram data.

In LBP features, every LBP value is in range from 0 to 255.

Next we select a range of intensity, say 5,

now we can make 255/5 = 51 +1 = 52 number of bars in histogram (from 0 to 51)

each bar will represent frequency of intensity in a block in its corresponding intensity range.

| Bar sl no | Intensity range | Bar sl no | Intensity range |
|---|---|---|---|
| 0 | 0-4 | 26 | 130-134 |
| 1 | 5-9 | 27 | 135-139 |
| 2 | 10-14 | 28 | 140-144 |
| 3 | 15-19 | 29 | 145-149 |
| 4 | 20-24 | 30 | 150-154 |
| 5 | 25-29 | 31 | 155-159 |
| 6 | 30-34 | 32 | 160-164 |
| 7 | 34-39 | 33 | 165-169 |
| 8 | 40-44 | 34 | 170-174 |
| 9 | 45-49 | 35 | 175-179 |
| 10 | 50-54 | 36 | 180-184 |
| 11 | 55-59 | 37 | 185-189 |
| 12 | 60-64 | 38 | 190-194 |
| 13 | 65-69 | 39 | 195-199 |
| 14 | 70-74 | 40 | 200-204 |
| 15 | 75-79 | 41 | 205-209 |
| 16 | 80-84 | 42 | 210-214 |
| 17 | 85-89 | 43 | 215-219 |
| 18 | 90-94 | 44 | 220-224 |
| 19 | 95-99 | 45 | 225-229 |
| 20 | 100-104 | 46 | 230-234 |

| | | | |
|---|---|---|---|
| 21 | 105-109 | 47 | 235-239 |
| 22 | 110-114 | 48 | 240-244 |
| 23 | 115-119 | 49 | 245-249 |
| 24 | 120-124 | 50 | 250-254 |
| 25 | 125-129 | 51 | 255 |

It is to be noted that the last bar (i.e. sl no 51) got only 255
so, now there will be 52 histogram features for each block,
Total will be 52x10x10 = 5,200 features for an image.
To calculate histogram data for an image the algorithm will be:

```
begin
Step 1: initialize an 3d array (say bar) of size 10, 10, 52 and assign 0 to every location
        (i.e. = bar[10][10][52])
Step 2: for each block_row_no in block in all rows  do:
step 3:         for each block_col_no  in block_row_no do:
step 4:                 for each lbp_val in LBP features within the selected block do:
step 5:                         bar[block_row_no] [block_col_no] [lbp_val/5]++
step 6                  end for
step 7          end for
step 8  end for
end
```

For example, the histogram data for the following 8x8 LBP features as follows:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 200 | 100 | 0 | 5 | 255 | 45 | 78 | 120 |
| 99 | 255 | 189 | 156 | 222 | 243 | 24 | 64 |
| 32 | 46 | 245 | 56 | 245 | 26 | 2 | 1 |
| 57 | 37 | 120 | 153 | 12 | 75 | 47 | 46 |
| 74 | 36 | 88 | 246 | 24 | 67 | 24 | 16 |
| 183 | 166 | 76 | 210 | 88 | 46 | 53 | 84 |
| 36 | 63 | 37 | 3 | 78 | 35 | 45 | 84 |
| 36 | 99 | 34 | 253 | 255 | 0 | 25 | 45 |

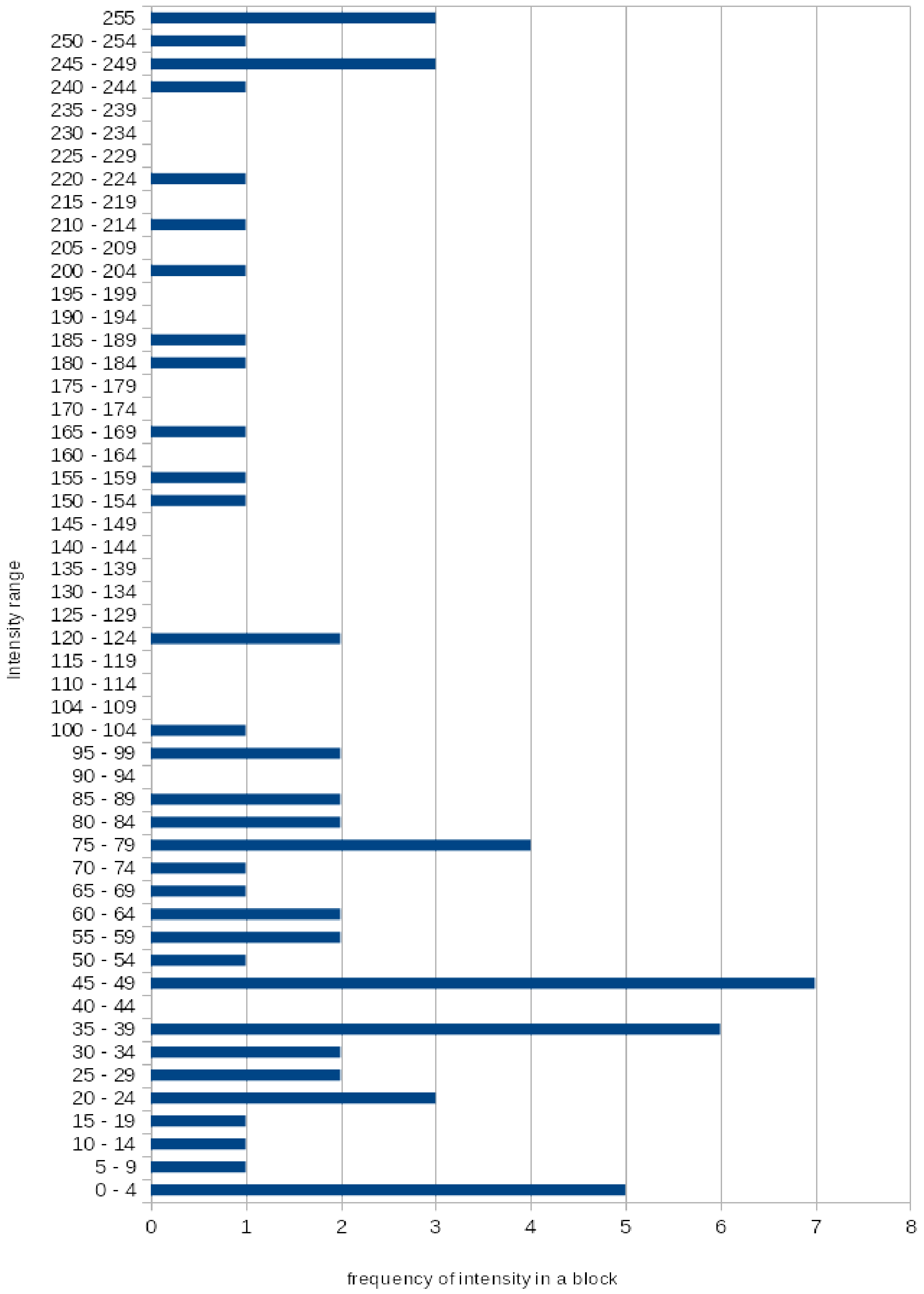


Fig: Histogram of the above LBP features

**3.4. Calculating Euclidean distance:**

The euclidean distance of any two value is computed as

eu_distance = $\sqrt{(val_1 - val_2)^2}$

For two 1D arrays $A_1$ and $A_2$ with n number of values is computed as:

eu_distance = $\sum_{i=1}^{n} \sqrt{(A_1[i] - A_2[i])^2}$

Now, we have Histogram data of images, contains 5,200 features per image. Now to find euclidean distance between histogram data of two images, the algorithm is:

```
begin
step 1: initialize eu_distance = 0
step 2: for each block_row_no in block in all rows  do:
step 3:         for each block_col_no  in block_row_no do:
step 4:                 for each $h\_val_1$ in histogram features within the selected block of
                        $image_1$ and $h\_val_2$ in histogram features within the selected block
                        of $image_2$ do:
step 5:                         eu_distance = eu_distance + sqrt(square($h\_val_1$ – $h\_val_2$))
step 6:                 end for
step 7:         end for
step 8: end for
end
```

**3.5. Producing Result:**

The purpose of the system in this project is to identify a person from face image using already stored minimum number of face images.

The euclidean distance can now tell the difference of images, higher the euclidean distance lower the chance of match, and lower the euclidean distance higher the chance of match.

If we have n number of images stored and we want to identify a person face, then there will be n number of euclidean distances between image to be tested and stored images. Now the stored image with lowest euclidean distance will be the result, and may be the person to be identified. This  is done when there is only one image stored per person.

But when there is more than one image stored per person, then euclidean distance between one person images with tested image will be sum of all euclidean distances between each images of that person with the tested image. Then the minimum euclidean distance will be the resulted person to be identified. The algorithm is as follows :

```
begin
step 1: initialize min_eu = +inf, eu_distance = 0,resulted_person = person[0]
step 2: for each person p in person_images_database:
step 3:         for each image i in p do:
step 4:                 eu_distance = eu_distance + euclidean_distance(i,test_image)
step 5:         end for
step 6:         if min_eu > eu_distance then
step 7:                 min_eu = eu_distance
step 8:                 resulted_person = p
step 9:         end if
step 10:        eu_distance = 0
step 11: end for
end
```

But sometimes, the matched person is not correct. This happens with a less differences of euclidean distances. To make it better, 2 other possibilities of matching result is computed. This is done by simply sorting all euclidean distances in ascending order and took the $2^{nd}$ and $3^{rd}$ results as other possibilities.

Computing accuracy of recognition:

This is computed simply by dividing number of matched by number of tests and multiplied by 100 to produce percentage.

$$\text{Accuracy} = \frac{number\ of\ match}{number\ of\ tests} \times 100$$

The number of match is computed by matching the test image label and resulted person name.

For this test training datasets have person with different name. I have taken 3 databases, one is friends database which contains 7 persons and 5 images per person, this database is created by myself.

Next one is ORL database, which contains 40 persons and 10 images per person. The images of this database have variant poses.

ORL face database source from :

http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

thanks to AT&T Laboratories Cambridge.

And the 3rd database is faces94 which contains 151 individuals and 20 images per person.

These databases have male and female persons faces.

Faces94 source from: http://cswww.essex.ac.uk/mv/allfaces/faces94.html

For the testing purpose from each individual one image is moved to other directories and stored in a  directory with a label as the directory name.

# CHAPTER 4. IMPLEMENTATION

## 4.1 LBP function

The following function reads an image from location ‘str’ and then calculate it’s LBP and store it in an array ‘ar[]’ by sending arguments starting row=i,ending row = row, starting column  = v, ending column = col

```
void lbp(unsigned char *ar,string str,int i,int v,int row,int col){
        int k = 0;
        Mat img = imread(str,CV_LOAD_IMAGE_GRAYSCALE);
        for ( ;i<row;i++) {
                for(int j = v ;j<col;j++) {
                        int t = 0;
                        t = img.at<uchar>(i-1,j-1) >=  img.at<uchar>(i,j)   ? (t<<1)|1:t<<1;
                        t = img.at<uchar>(i-1,j)   >=  img.at<uchar>(i,j)   ? (t<<1)|1:t<<1;
                        t = img.at<uchar>(i-1,j+1) >=  img.at<uchar>(i,j)   ? (t<<1)|1:t<<1;
                        t = img.at<uchar>(i,j+1)   >=  img.at<uchar>(i,j)   ? (t<<1)|1:t<<1;
                        t = img.at<uchar>(i+1,j+1) >=  img.at<uchar>(i,j)   ? (t<<1)|1:t<<1;
                        t = img.at<uchar>(i+1,j)   >=  img.at<uchar>(i,j)   ? (t<<1)|1:t<<1;
                        t = img.at<uchar>(i+1,j-1) >=  img.at<uchar>(i,j)   ? (t<<1)|1:t<<1;
                        t = img.at<uchar>(i,j-1)   >=  img.at<uchar>(i,j)   ? (t<<1)|1:t<<1;
                        ar[k++] = t;
                }
        }
}
```

## 4.2 Histogram function

This function calculates histogram features from lbp features from ar[] and stored it in his[], where pixels is the size of the array ar[], size of ar[] and his[] are same.

```
void histogram(unsigned char *ar,unsigned char *his,int pixels){
 for (int i = 0;i<pixels;i++)
        his[ar[i]/5]++;
}
```

## 4.3 Euclidean function

This following function return the euclidean distance between values in two arrays a1[] and a2[] where size is the size of those array.

```
double g_his_euclidean_distance(unsigned char a1[],unsigned char a2[],int size){
        double result = 0;
        for (int i=0;i<size;i++)
                result+=pow(a1[i]-a2[i],2);
        return sqrt(result);
 }
```

## 4.4 Accuracy measure

Following statement will compute accuracy of the system and assign it to accuracy in terms of percentage. Where test_count is the total number of tests and false_count is total number of wrong output.

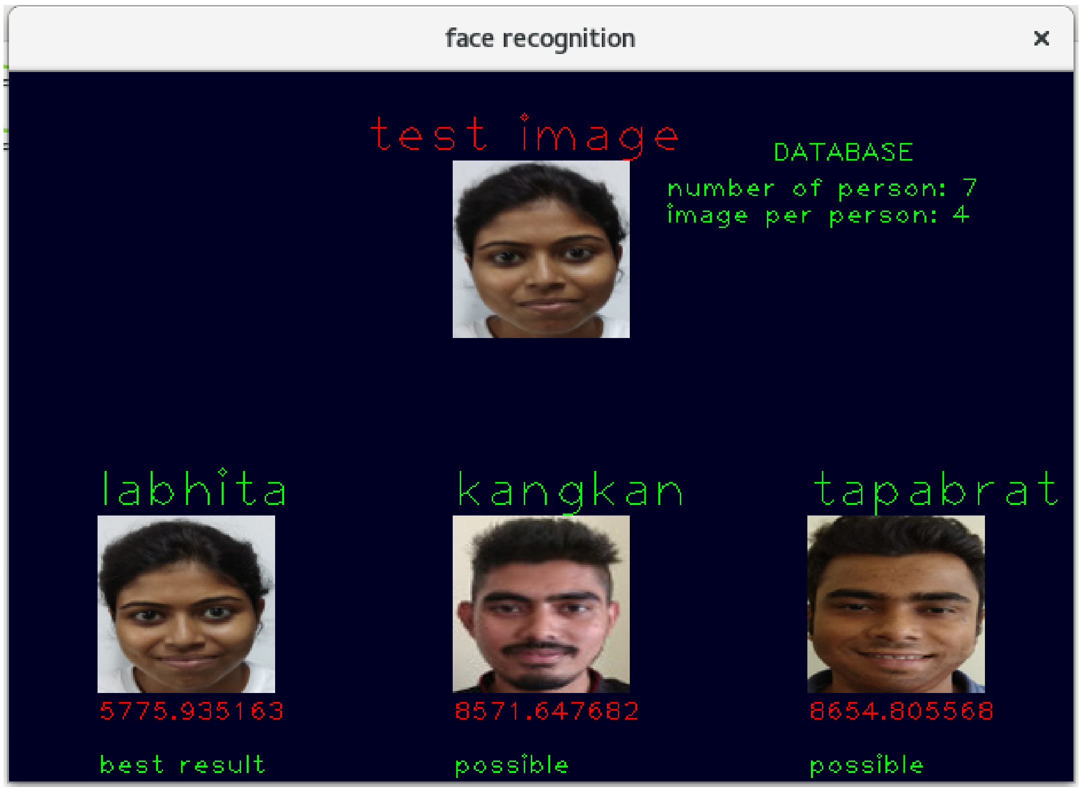
```
double accuracy = ((double)(test_count-false_count)/test_count)*100;
```

# CHAPTER 5. OUTCOMES

**A**ccuracy of recognition:

| Database | Individuals | Image per person | Number of test images | Miss | Match | Accuracy % |
|---|---|---|---|---|---|---|
| ORL | 30 | 1 | 30 | 11 | 19 | 63.333 |
| | | 2 | 30 | 8 | 22 | 73.333 |
| | | 3 | 30 | 7 | 23 | 76.6667 |
| | | 4 | 30 | 9 | 21 | 70 |
| | | 5 | 30 | 8 | 22 | 73.333 |
| | | 6 | 30 | 7 | 23 | 76.6667 |
| | | 7 | 30 | 7 | 23 | 76.6667 |
| | | 8 | 30 | 5 | 25 | 83.3333 |
| | | 9 | 30 | 3 | 27 | 90 |
| | 40 | 1 | 40 | 18 | 22 | 55 |
| | | 2 | 40 | 14 | 26 | 65 |
| | | 3 | 40 | 13 | 27 | 67.5 |
| | | 4 | 40 | 14 | 26 | 65 |
| | | 5 | 40 | 12 | 28 | 70 |
| | | 6 | 40 | 12 | 28 | 70 |
| | | 7 | 40 | 11 | 29 | 72.5 |
| | | 8 | 40 | 10 | 30 | 75 |
| | | 9 | 40 | 8 | 32 | 80 |
| faces94 | 151 | 19 | 151 | 1 | 150 | 99.337 |
| | | 10 | 151 | 3 | 148 | 98.0132 |
| | | 4 | 151 | 2 | 149 | 98.6755 |
| Friends | 7 | 4 | 12 | 3 | 9 | 75 |

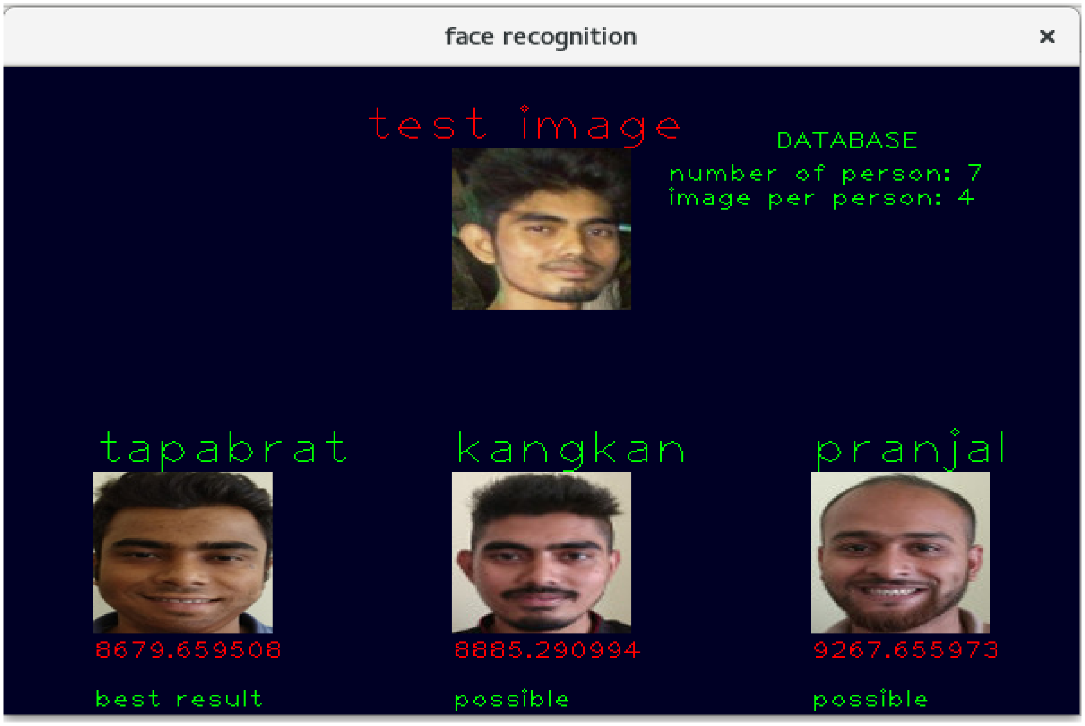
These are the screen shots of the program resulting  unknown test images:



```
face recognition

test image                DATABASE
                          number of person: 7
                          image per person: 4

labhita          kangkan          tapabrat
5775.935163      8571.647682      8654.805568
best result      possible         possible
```



```
face recognition

test image                DATABASE
                          number of person: 7
                          image per person: 4

tapabrat         kangkan          pranjal
8679.659508      8885.290994      9267.655973
best result      possible         possible
```

# CHAPTER 6. CONCLUSION AND FUTURE WORK

## 6.1 Conclusion:

The purpose of this project is to build a face recognition system system based on local binary pattern features using less number of faces per person. From the result of this project/program it is true that more the number of faces per person more the recognition accuracy.

Also more the execution time. But for the less number of faces per person LBP is known to be the best features till now. Also the accuracy is depend on the classifier that is used. Here euclidean distance is working as a classifier, but SVM(support vector machine) will give more accuracy and lot more faster. The library used in the program is OpenCV which gives automatic optimization by using intel IPP library but not for every task. So, to achieve this intel IPP library can be used and manually instructions of IPP can be used, but also intel IPP will work on intel processor most of the time. There are built in functions in OpenCV that could be used, but I have implemented some manually to learn about them.

If an unknown test face is given to the program and the corresponding person is not in the database, then still the program will result some faces, which is incorrect. To resolve this issue a threshold of euclidean distance can be considered and if the distance is minimum than threshold then results possible faces.

Also I have used the simplest original LBP algorithm, but there are more variation of LBP, which may results more accuracy.

## 6.2 Future work:

**F**ace detection: in this project I have extracted faces from images using third party tools, also resize them. But this could be done automatically by adding another sub system to the project. Face detection can be done using LBP, but we should choose a good classifier and more images of face and non face, and for the classifier SVM will work fine, and will be good is there is less number of training images.

Face recognition from video : in this project recognition is done from images, also it can be done in video by taking frames images and then detect faces and then feed it to the program. But using euclidean distance  will result very slow recognition, to solve this a faster classifier can be used, like SVM.

Recognition is other image: I have used human faces for recognition, but LBP can be used in many types images, like texture recognition and also other object recognition.

## REFERENCES:

[1] T. Ojala, M. Pietikäinen, and D. Harwood (1994), "Performance evaluation of texture measures with classification based on Kullback discrimination of distributions" Proceedings of the 12th IAPR International Conference on Pattern Recognition (ICPR 1994), vol. 1, pp. 582 – 585.

[2] “Face Recognition from a Single Image per Person: A Survey”
Xiaoyang Tan, Songcan Chen ,Zhi-Hua Zhou,Fuyan Zhang
Department of Computer Science and Engineering
Nanjing University of Aeronautics & Astronautics, Nanjing 210016, China
National Laboratory for Novel Software Technology
Nanjing University, Nanjing 210093, China
State Key Laboratory of Pattern Recognition
Institution of Automation, Chinese Academy of Sciences, Beijing 100080, China

[3] “One Sample Per Person Facial Recognition With  Local Binary Patterns and Image Sub Grids”, Gordon Stein, Yuan Li, Dr. Yin Wang
Department of Math and Computer Science
Lawrence Technological University Southfield, Michigan, USA ,2016 Annual Conference on Information Science and Systems (CISS)

[4] “Efficiency of Recognition Methods for Single Sample per Person Based Face Recognition”
Miloš Oravec, Jarmila Pavlovičová, Ján Mazanec, Ľuboš Omelina, Matej Féder and Jozef Ban
Faculty of Electrical Engineering and Information Technology
Slovak University of Technology in Bratislava,
Slovakia